Program 6 - Bulldog Design Pattern Implementation

Author: Abigail Pitcairn
Version: March 26, 2025

Code: https://github.com/abbyPitcairn/BullDog

Task 1

I gave the AI tool the text from the assignment for this task and it output PlayerListModel.java, a class which included a private ArrayLists of Players and Observers and methods to add players, get player's names and scores, set player's score, add observers, and notify observers. I moved this class into my Eclipse project and it underlined an error where there was a call to an object called ModelObserver; this type of object was not defined in the project:

```
/**
 * Adds an observer to the model.
 * @param observer The observer to be added.
 */
public void addObserver(ModelObserver observer) {
    observers.add(observer);
}

/**
 * Notifies all registered observers about changes in the model.
 */
private void notifyObservers() {
    for (ModelObserver observer : observers) {
        observer.modelChanged();
    }
}
```

I asked the AI tool to generate the ModelObserver class to accompany the PlayerListModel class, which it did, and when I copied this new class into the project, there were no more errors.

```
/**
 * Interface for objects that observe changes in the model.
 * This is part of the Observer pattern and is used to notify views or controllers
 * whenever the model updates.
 */
public interface ModelObserver {
    /**
     * Called when the observed model changes.
     */
    void modelChanged();
}
```

This was all I did before moving on to the next tasks. During task 2, some code was generated that called a method, playerListModel.getPlayers(), which threw an error because no such method existed. I prompted the AI tool to fix this by adding such a method to PlayerListModel.java, which it did correctly on the first prompt:

```java
/**
 * Returns the list of players.
 * @return The list of players.
 */
public ArrayList<Player> getPlayers() {
    return new ArrayList<>(players);
}
```

Task 2

I gave the AI tool the text from the assignment for this task. The output was the ScoreboardViewer class, which extends JPanel and implements the ModelObserver class. It had a private PlayerListModel and a private JPanel, a ScoreboardViewer constructor, an override of the modelChanged() method to call the class's only other method, updateScoreboard().

I moved this code into the project and found an error underlined in red in the call to getPlayers() in the PlayerListModel class, a method which did not yet exist. I had the AI tool edit PlayerListModel to fix this, as I previously described. The AI also added a main method to the ScoreboardViewer class, which was manually removed because I wanted to test everything all together in the following integration task.

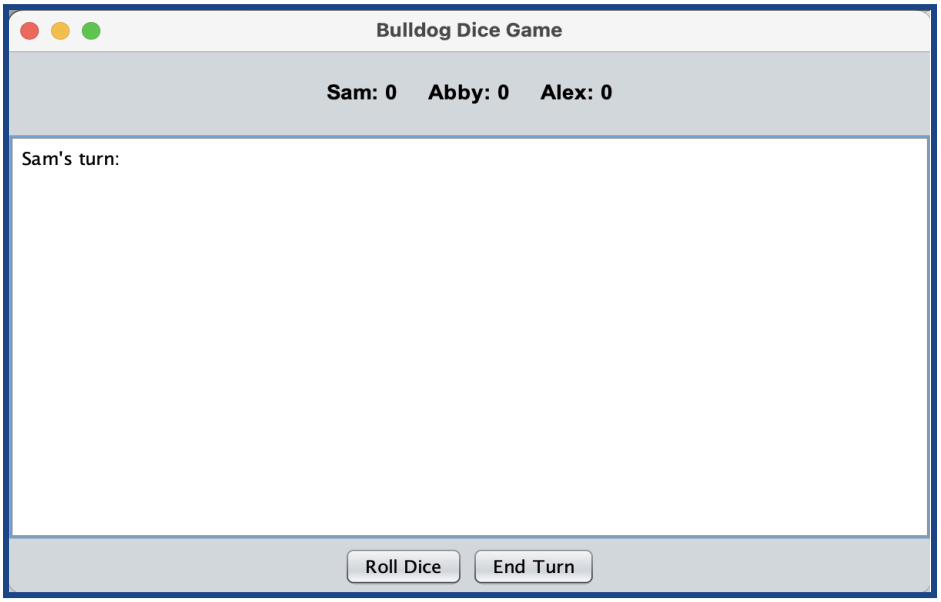
After integrating the new features into the GUI and running it, I realized that the score panel was a little off; it was displaying the player's names and scores in a vertical line instead of horizontal. I neglected to screenshot the original format, but ChatGPT's original scoreboard looked something like this:

Abby: 0
Sam: 0

Whereas I wanted it to look more like how it was in the previous iterations of this program:

Abby: 0 Sam: 0

I gave ChatGPT a prompt to change this formatting, providing it with the same examples as above to visually demonstrate what I wanted it to look like, and it changed the layout from Grid to Flow. I moved this new ScoardBoardViewer code into my project, ran it, and it was correctly formatted:



ChatGPT's scoreboard design included bolded text with an inline comment that said this was for "improved visibility"; I liked this, so I decided to keep it, even though it was not something I explicitly asked for.

Task 3

I jumped around a little bit in this project, as I found it was faster to integrate the new features into the GUI and test everything all together rather than to write individual tests for the features before integrating. This back and forth may cause a bit of repetition in my lab book here, but I wanted to discuss the process as a whole, as well as have notes that elaborated on previous tasks included in the section on that task.

The first prompt I gave ChatGPT for task 3 was the text from the assignment, without including the debugging notes. I received a new copy of BulldogGUI.java which was based on program 4 (I guess the last official version I had given ChatGPT of this class was from program 4, as it had "package program4;" on the first line) which I moved into the program 6 project. The only manual edit I made at this point was changing that first line to the correct package name.

Then, I read through the program to check for any other errors. The first red underline was under a call to Player.play(), which was missing an argument of a Dice object. I had copied over Dice.java from the previous assignment, but ChatGPT had not implemented it yet into the GUI. I prompted the AI to add a Singleton Dice object and provided it with the contents of Dice.java for reference. I moved this newly generated BulldogGUI.java into my project. This is what ChatGPT added across the GUI class; first in the BulldogGUI constructor, then in the rollDice() method for Human Players, and lastly in the nextTurn() method for non-Human Players:

```
dice = new Dice();
```

```
int roll = dice.roll();
```

```
turnScore = currentPlayer.play(dice);
```

This seemed consistent with the Singleton pattern, as only one Dice was going to be created per GUI, and the same object was used throughout.

The next red underline was under playerListModel.setScore(...) because PlayerListModel did not have a method named setScore; it was actually called setPlayerScore. I manually changed the name, and didn't need to change anything else because ChatGPT had gotten the parameters correct, just not the method name. After that, all red underlines were gone and I ran the program.

The program failed to run, giving an error message that playerListModel could not addObserver() because it was null. I prompted ChatGPT to "Fix the BulldogGUI class based on this error message:" and copied and pasted the full error into the prompt. ChatGPT recommended moving the initialization of the

playerListModel to before the scoreboardViewer was created, and provided the updated code to do so. I moved this code into my project and ran it again.

This time, there was a different error message, stating that a JFrame window was being added to a container, which is not allowed. I configured the same prompt to ChatGPT with this new error message, and it gave the corrected code for the BulldogGUI constructor. However, running this “corrected code” threw the same error. I read back ChatGPT’s message and after providing the updated code for the GUI, it said to “ensue that scoreboardViewer was a panel component, not a window”, so I gave ChatGPT the contents of the ScoreboardViewer class and asked it to check for that. It gave me the corrected code for the class, which I moved into my project, ran it, and now the game ran successfully.

It wasn’t until this point that I actually noticed the scoreboard formatting was off, and I went back and had ChatGPT reformat it as I described at the end of task 2. I did some further testing with different numbers of players and different player types to ensure everything was working correctly. I did a couple manual edits, such as disabling the “Continue” button when the game ended (this was the only one of the three buttons left enabled, and it was such a quick fix it would’ve taken longer to have ChatGPT do it) and adding some Javadoc comments to methods at the end of BulldogGUI.java that ChatGPT had missed.

Summary and Notes

ChatGPT did really well on this program. I didn’t run into any major issues or run into any circular prompting nightmares (what I mean by this is, at no point did I have ChatGPT tell me something wrong over and over, which it sometimes likes to do). It seemed clear the AI is familiar with these design patterns that were needed for this project, and it knew how to implement them.

I am not entirely sure if ChatGPT got this program 100% correct, as I am not an expert on these design patterns yet. But from what I can tell, it did a really good job, especially in the integration of the new features for task 3. It was effective working in small steps to correct errors iteratively, rather than having a bunch of confusing errors all at once, and it felt almost too lucky that ChatGPT was able to correct all the errors on its own.

Overall, it took about 40 minutes of prompting the AI and 35 minutes of manual editing (not including the time spent proofreading, as that’s not really a necessary step, but rather just something I do). Additionally, it did not take very much “expertise” from myself, as the manual edits were more surface level stuff, rather than big complex ideas.